Systems

Items-Inventory

The `InventoryManager` uses 4 different classes:

- `InventoryItem`
  A scriptable object class that contains everything a generic item would need.
- `ClothingItem`
  A scriptable object class inheriting from `InventoryItem` that adds clothing specific variables.
- `ItemObject`
  A MonoBehaviour class inheriting from `Interactable` that is attached to the item prefab of an `InventoryItem`.
- `InventorySlot`
  A class that stores an `InventoryItem` and the stack size.

The `InventoryManager` makes use of a dictionary that works in parallel with the inventory making it so you can use `InventoryItem` as keys and `InventorySlot` as values.

`InventorySize` returns the total amount of items you have in your inventory.

`AddItem` adds the given item to the dictionary and inventory if the player has enough space and it makes sure the stack size gets increased if that item is already present.

`RemoveItem` removes the given item or the given inventory index in the dictionary and inventory and it makes sure the stack size gets decreased if that item is still present or it gets completely removed.

`UpdateIventoryUI` loops over every UI inventory slot and makes sure the values are up to date.

`PressedSlot` gets used by the UI Sell Button, if a valid slot is pressed and that slot contains a requested item it removes the item from the `ShopkeeperManager` requested items and removes it in the players inventory.

Shopkeeper

The `ShopkeeperManager` uses 3 different classes:

- `InventoryItem`
- `InventorySlot`
- `ShopkeeperInteraction`
  A class containing variables about a shopkeeper interaction like dialog text or item requests.

`Interact` handles all the shopkeeper interactions and constantly counts up with every interaction, then it looks that the list of interactions and executes the next one, it uses 3 interaction types (dialog, request and Random Request) to determine what method to use.

`UpdateRequestUI` removes all the request object UI and instantiates only the ones that are left.

`RemoveItem` removes the given item and it makes sure the stack size gets decreased if that item is still present or it gets completely removed, it also makes sure it goes to the next interaction if all the requested items are removed.

`IsRequestedItem` returns true if the given item is a requested item, and returns false if not.

`Request` initializes a new request if not already done and makes sure you can't skip the request by interacting again.

`Dialog` is a `IEnumerator` that loops over every character in a string and displays them 1 by 1 with a delay in between.

Player

The `PlayerController` uses 2 different classes:

- `ClothingItem`
- `Interactable`
  A MonoBehaviour class containing a virtual method called Interact.

`UpdateMoneyText` changes the money text to the updated current money text.

`UpdateAnimators` sets the floats of the shirt, shoe, and player animators.

`UpdateClothing` Deletes all the clothing and instantiates the new ones while syncing the animations.

Closet

`ClosetManager` is a class inheriting from `Interactable` that handles part of the UI for the closet menu and interacts with the player.

`ClosetSlot` is a MonoBehaviour class that is attached to each slot in the closet menu, it also handles the buttons of the UI.

`Buy` makes sure the player has enough money to buy that item and changes the button to equip next.

`Equip` assigns the equipped clothing type to the player and calls `UpdateClothing` and `UpdateIventoryUI`.

`Pressed` checks if the slot is bought and calls `Buy` or `Equip` depending on that.

Process

23-8-2022

I started setting up a GitHub repository with a standard 2D Unity project. Then I went searching for some assets. Because I'm not really an artist I just looked on itch.io and found a template character with some modular clothing. I changed this a bit so its black and white so I can change the colours of these clothes in Unity.

I also started to form an idea: The shopkeeper request some items you need to get out of the shop and you need to bring them to him, after that you get some cash. With this cash you can buy and equip clothing with gives you powerups (probably just more movement speed). So you can get the items quicker and make more money.

After the initial idea I started working on the player controller. I wanted to keep it simple so I did, but I think it still feels like it has complexity to it because of the acceleration and dampening. After this was done I wanted to add the animations, for both the clothing and the player character. This was very easy to setup but it took a bit of time. I had an idea for a more complex solution to animating the clothing on the player by making a custom implementation for animating but because of the lack of time for this prototype I stuck with the Unity animator. It was a little bit tedious setting up the animation tree but in the end it worked perfectly.

I changed the clothing so it was more expandable. Now the animations and “items” are bound to a scriptable object, so I can easily create more clothing with new sprites and animations(but in my case I will only change the colour).

After that I went on to search for some more assets, this time the environment and the items that I use in that environment. I found an interior pack with some different kinds of food in it so I started making something resembling a supermarket.

24-8-2022

The first thing I did was continue making the environment and adding the layering and collisions so the player can move seamlessly through the level without any Z-fighting.

After this I started making the inventory system, and after a bit of trial and error and finding some inspiration on the internet I got something that worked very good and was not that complex. I also made another scriptable object for the generic items so we can differentiate between clothing items and normal items.

I also got assets for the UI, so I started working on the inventory UI but I didn’t use them because I don’t think it fits well. Because of the scale of this game I didn’t find it necessary to implement mouse control with the inventory (switching slots and dragging/dropping).

I continued with the inventory system and made it so you can actually see the items in your inventory with the amount. And I added the text to the Icons to indicate the amount of items that you have.

25-8-2022

I started with working on the Shopkeeper and all the systems that it would need. The interaction with the player and “locking” the player was the first thing and it was very easy to do. The second was making the dialog and request system. This was a lot more effort to do but it got completed without any hiccups.

Also making the items from the inventory interact with the shopkeeper wasn’t as easy as I thought, and I really started running into limitations on the inventory system. I couldn’t really rework the inventory system because of the time constraint so I just made it work in the best way I could.

The disposing and selling of items worked on this point but the money system is not in place yet.

26-8-2022

I started with the UI for the closet, and coming up with a system to buy and equip the clothing items that I made in the first day. After trying a few things I came up with a very simple system that works a lot better than the inventory interaction.

Now after buying the clothes you can equip them, then they are synced by playing the same animation on the same frame. This is a limitation of the sprites and I'm not really happy with the solution but I couldn't think of another way to fix this issue.

I also finished the money, selling and buying completely with every item having a value, for clothing this is used to buy, and for food its used to sell.

After that I created some extra clothing and made the game more "games like" by creating some dialog and telling the player what to do. I also made a main menu really quickly and changed some debug variables so I was ready to build.

Thoughts

It was a fun experience. It felt like a game jam with very specific limitations but instead of making a fun game as fast as possible I had to focus a bit more on the code quality and expandability. This was easier said than done because I still want to make a good game and with this limited time its hard to do both. I included all the required features but they are not completely were all my time went in to because I felt like I couldn't make a game with the required features as centre point. I still have a lot of things that I would have changed or spend more time on if I was given a few more days but in this time period I'm happy with the end product.